# Kernel-based Virtual Machine (KVM) for Itanium Architecture

Software and Solutions Group

*Fenghua Yu <fenghua.yu@intel.com>*

*April 16, 2007*

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

All dates provided are subject to change without notice.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries (regions).

*Other names and brands may be claimed as the property of others.

Copyright © 2007, Intel Corporation. All rights are protected.

# Agenda

- Overview
- KVM introduction
- KVM/x86 Architecture
- KVM/IPF Design
- Status
- Reference information
- Q&A

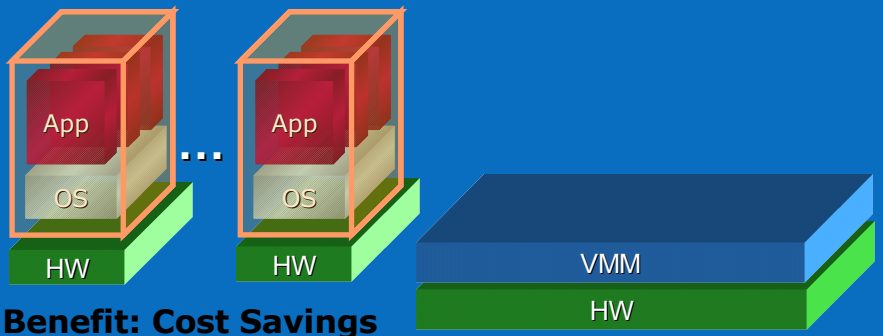
# Overview

- The following information is based on very early experiments and thoughts. Future design and information could be changed.
- Looking for input and feedback from the community.

# Virtualization Overview

Today

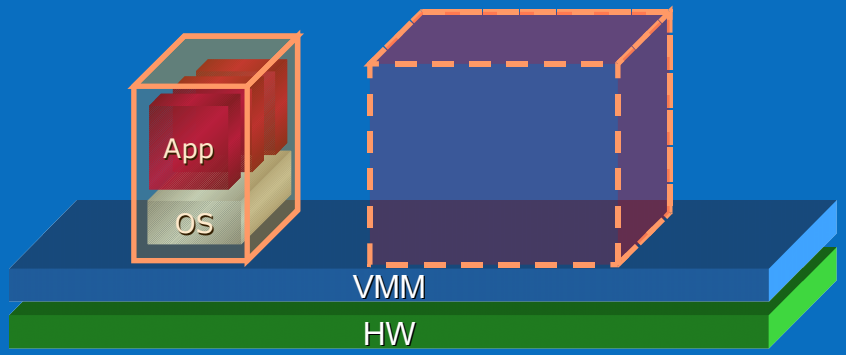
## Server Consolidation




**Benefit: Cost Savings**

- **Power and Cooling**
- **Hardware, Software, Management**

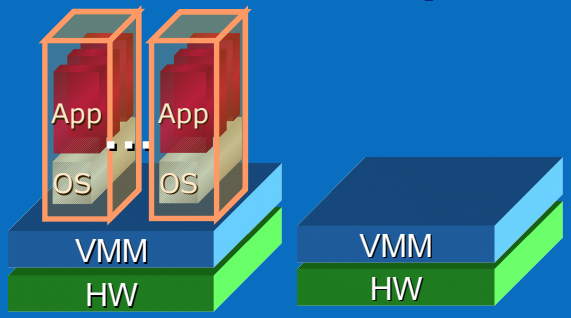
## R&D Production




**Benefit: Business Agility and Productivity**

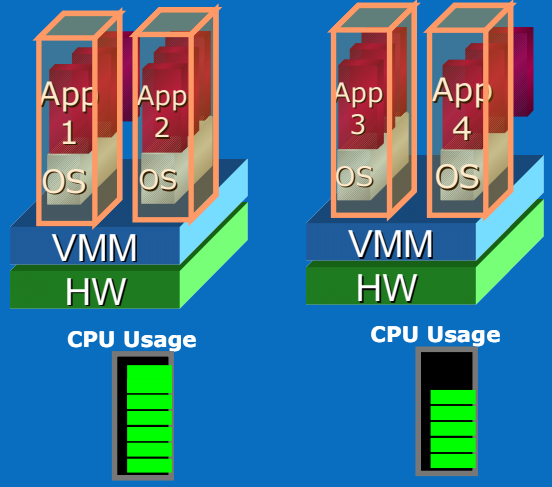
Emerging

## Disaster Recovery



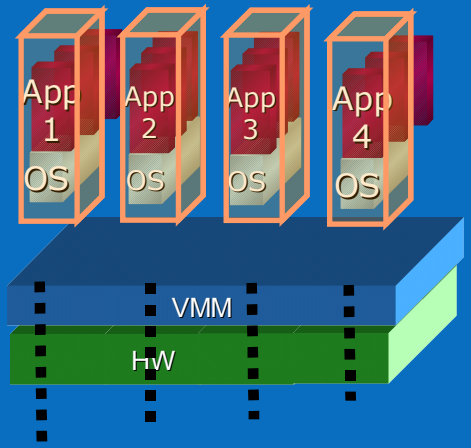

**Benefit: Business Continuity and Operational Efficiency**

## Dynamic Load Balancing



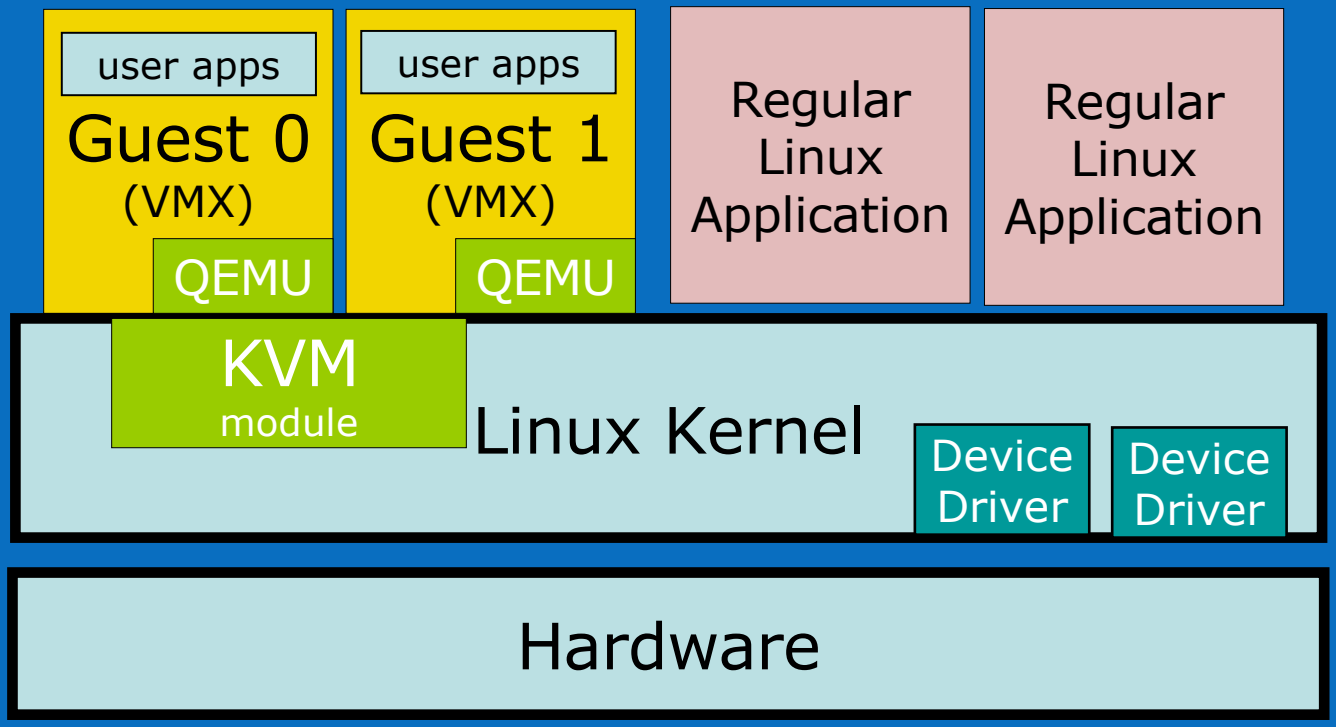

- **Benefit: Productivity**

## Partitioning



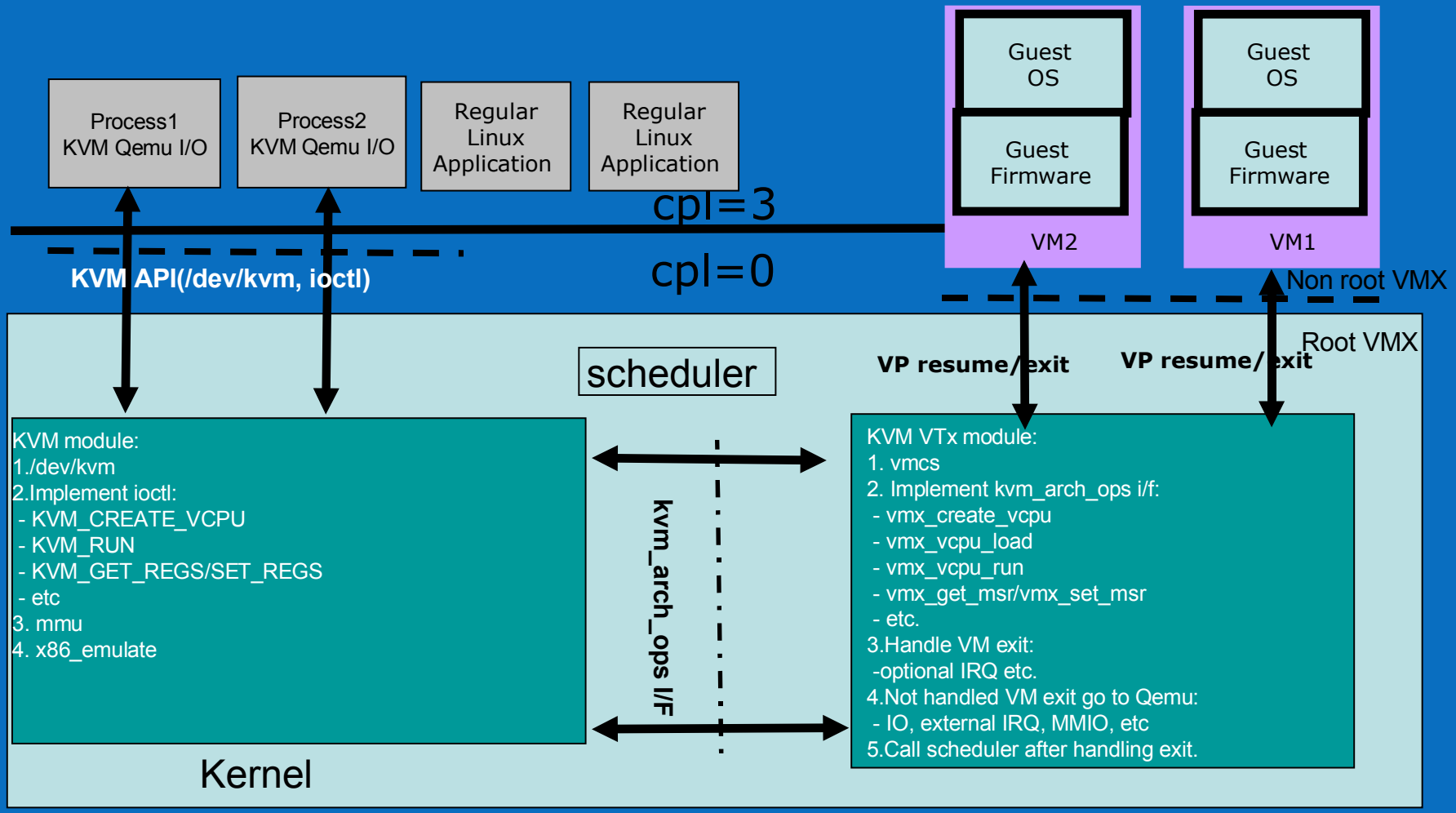

- **Benefit: Cost saving and Performance**

# KVM Introduction

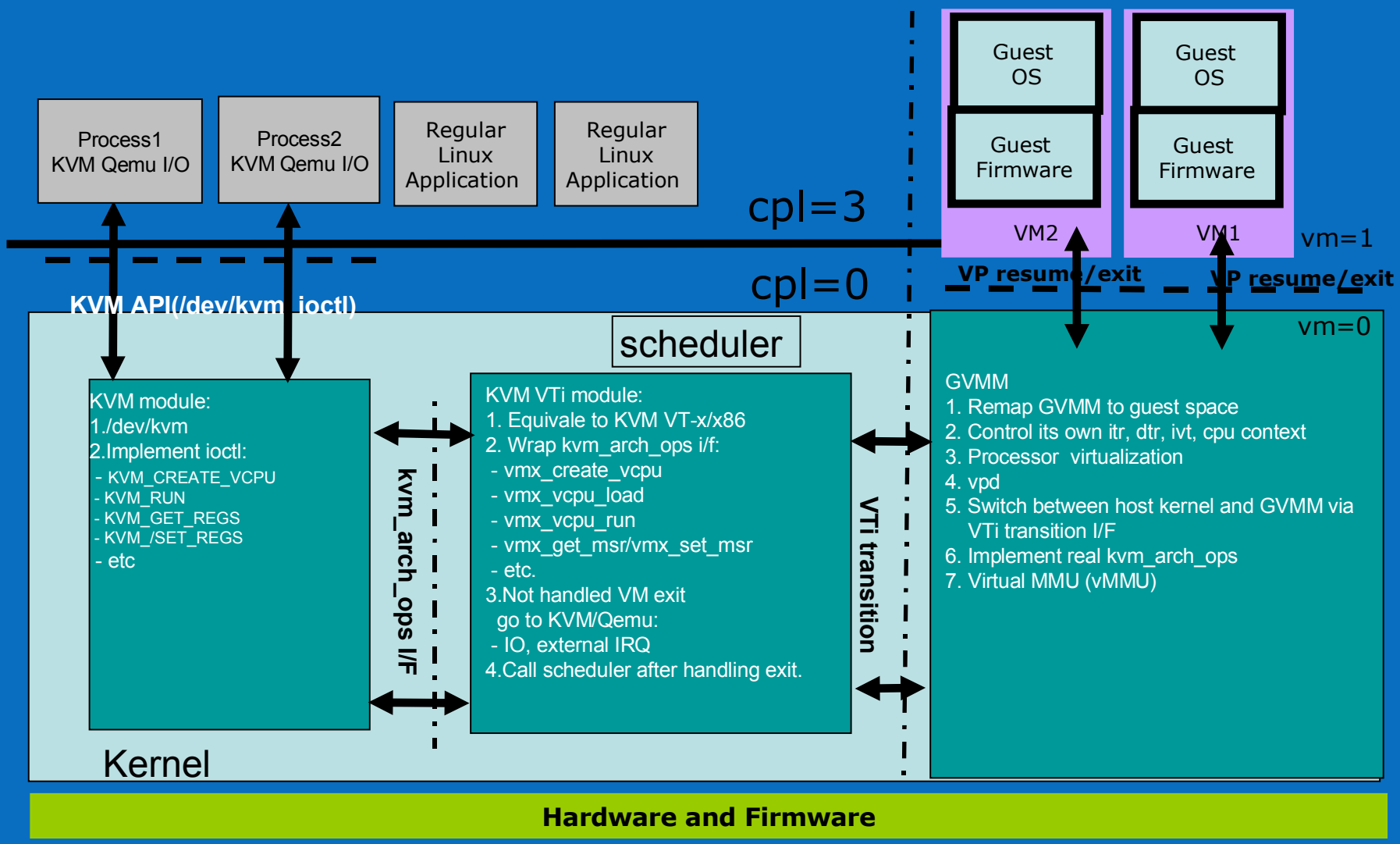



- In early development phase:
  - KVM was introduced in kernel 2.6.20
  - KVM works on X86 architecture and supports Intel VT-x and AMD SVM now, both 32 and 64.
  - Guest can be Linux or Windows.
  - Full virtualization only now
    - Paravirt is in progress.
  - Need more feature implementations:
    - SMP guest, opt MMU, multiple architecture support, optimization, etc.

# KVM/X86 VT-x Architecture

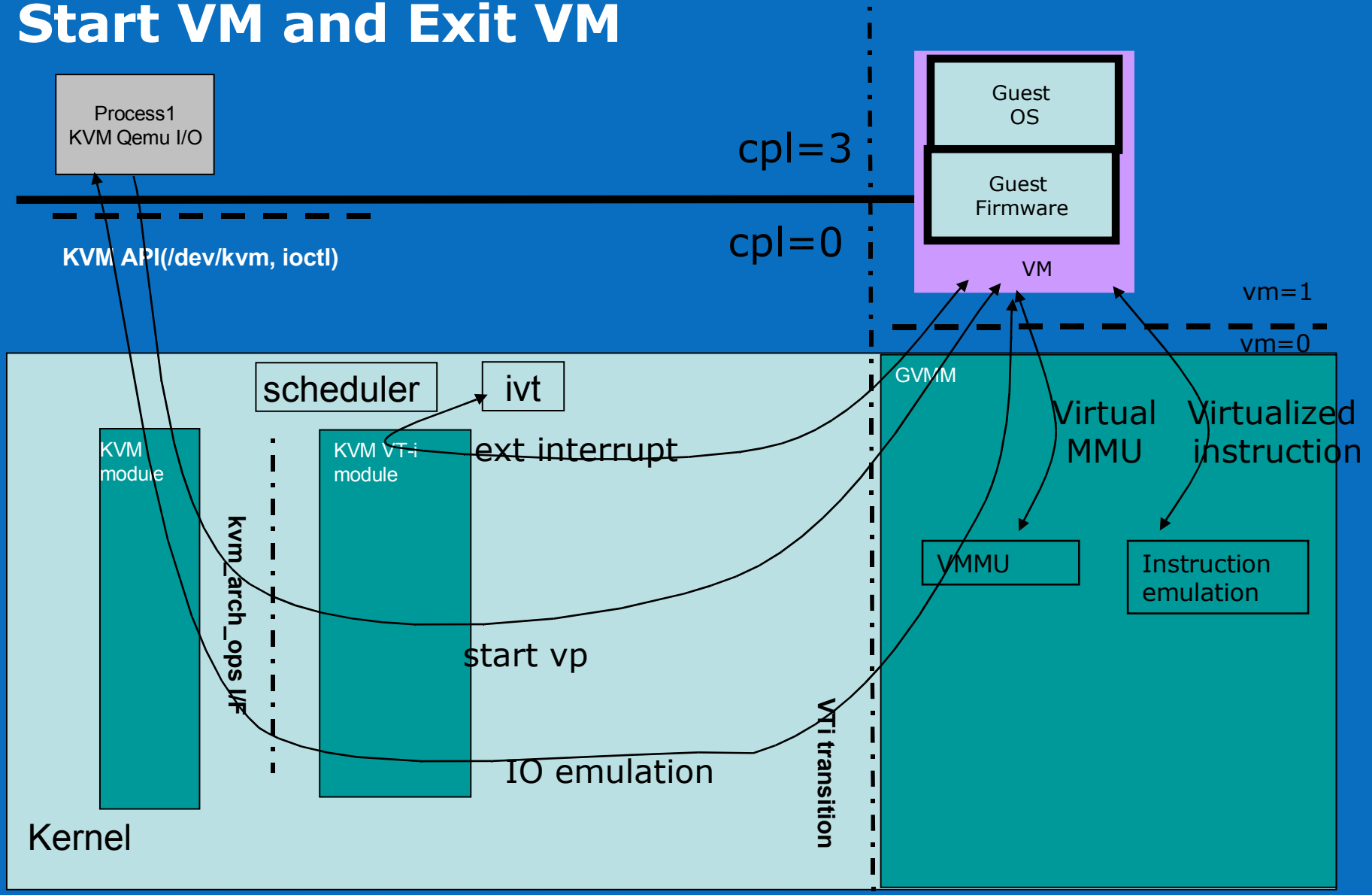


# KVM/Itanium VT-i Architecture

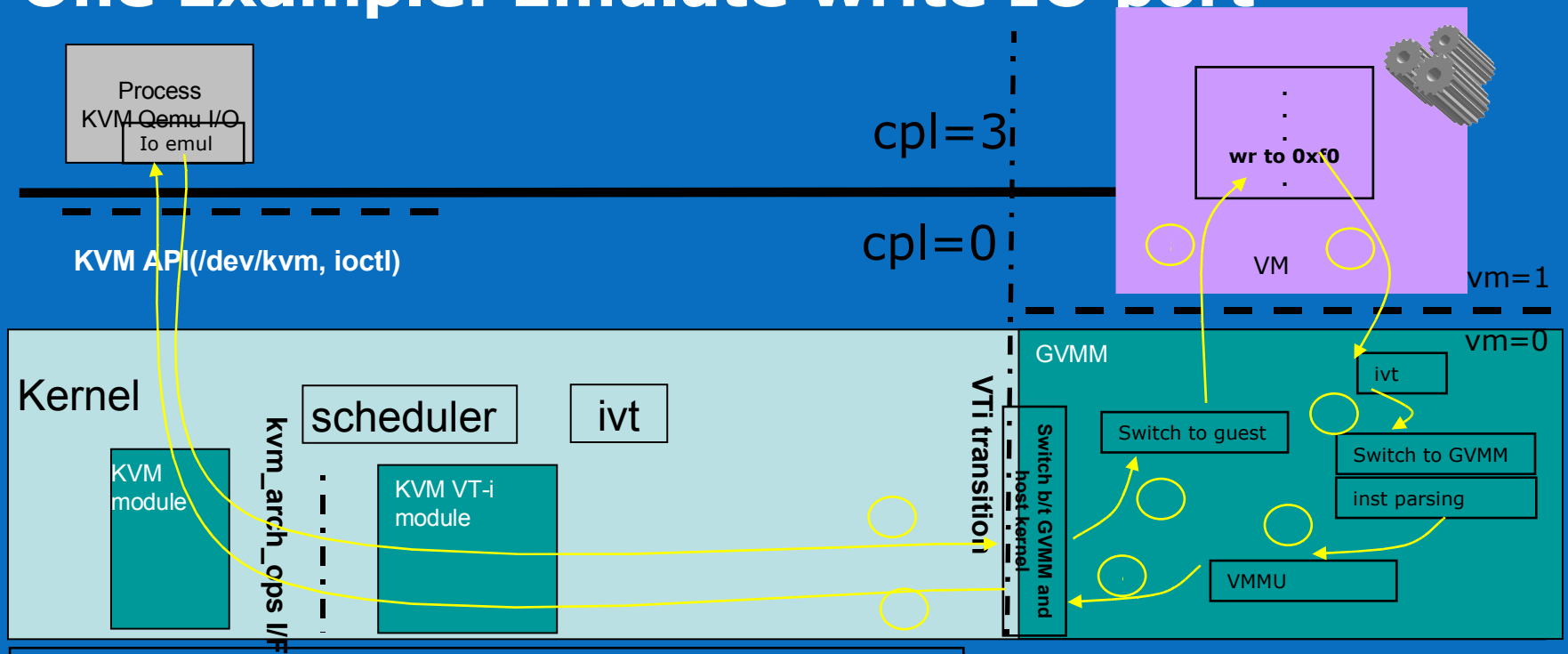



Considerations of GVMM: Guest Virtual Machine Monitor
1. VT-i VMM needs different ivt from host Linux
2. VT-i VMM has same address space as VM for performance

# Start VM and Exit VM



# One Example: Emulate write IO port




**Write I/O port emulation:**

1. VM executes w/r to 0xf0 and raises dtlb miss;
2. Save guest states and call trap handler to parse the fault instruction;
3. MMU memory address in MMIO and IO request is sent to Qemu;
4. Switch to host kernel mode via VTi transition;
5. Go back to Qemu to do 0xf0 port emulation;
6. Re-enter kernel and switch back to GVMM;
7. Restore guest states;
8. Execute next instruction in guest

# VMMU Design Issues

- RID handling
  - ?? Flush TLB between VT-i transition
    - Performance impact on both guest and host
  - ?? OR RID partitioning
    - For example, 18-23 bits in rid is used for rid partition id to distinguish guest and host rid.
    - Host uses rid partition id=0 while guest uses rid partition id=1~63
    - Host change, shortage of rid for each guest or host, limited number of guests.
- TR usage:
  - GVMM holds ITR and DTR for code and data.
    - Impact VTi transition code and VM performance
    - KVM VTi module does itr.i/itr.d before transition to GVMM.
    - KVM VTi module does ptr.i/ptr.d after transition from GVMM.
  - Problems for itr/ptr in modules
    - Conflict TR insert/purge among modules
    - itr can not be executed in tc mapped code (KVM VTi module) because
      - psr.ic=0 is required for itr
      - But TLB miss from tc mapped code will lose iip, ipsr, etc and can not be handled
  - Need host kernel interface for insert/purge TR.

# Status

- Initial prototype architecture design is done
- Prototype code is in progress.
- Near future target is to fully boot Linux and Windows.
- Well KVM coding for multi-architecture porting.
- Will keep alignment with mainline KVM features e.g. SMP, power management, performance tuning, etc.

**Need Help and Cooperation from the Community**

# Reference Information

- KVM wiki: http://kvm.qumranet.com/kvmwiki
- Xen: http://www.xensource.com/
- Intel VT-i specification: ftp://download.intel.com/technology/computing/vptech/30594201.pdf
- Intel VT-x specification: http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/pentium4/reference/197666.htm
- Intel Itanium Architecture Software Developer's Manuals: http://www.intel.com/design/itanium/manuals/iiasdmanual.htm

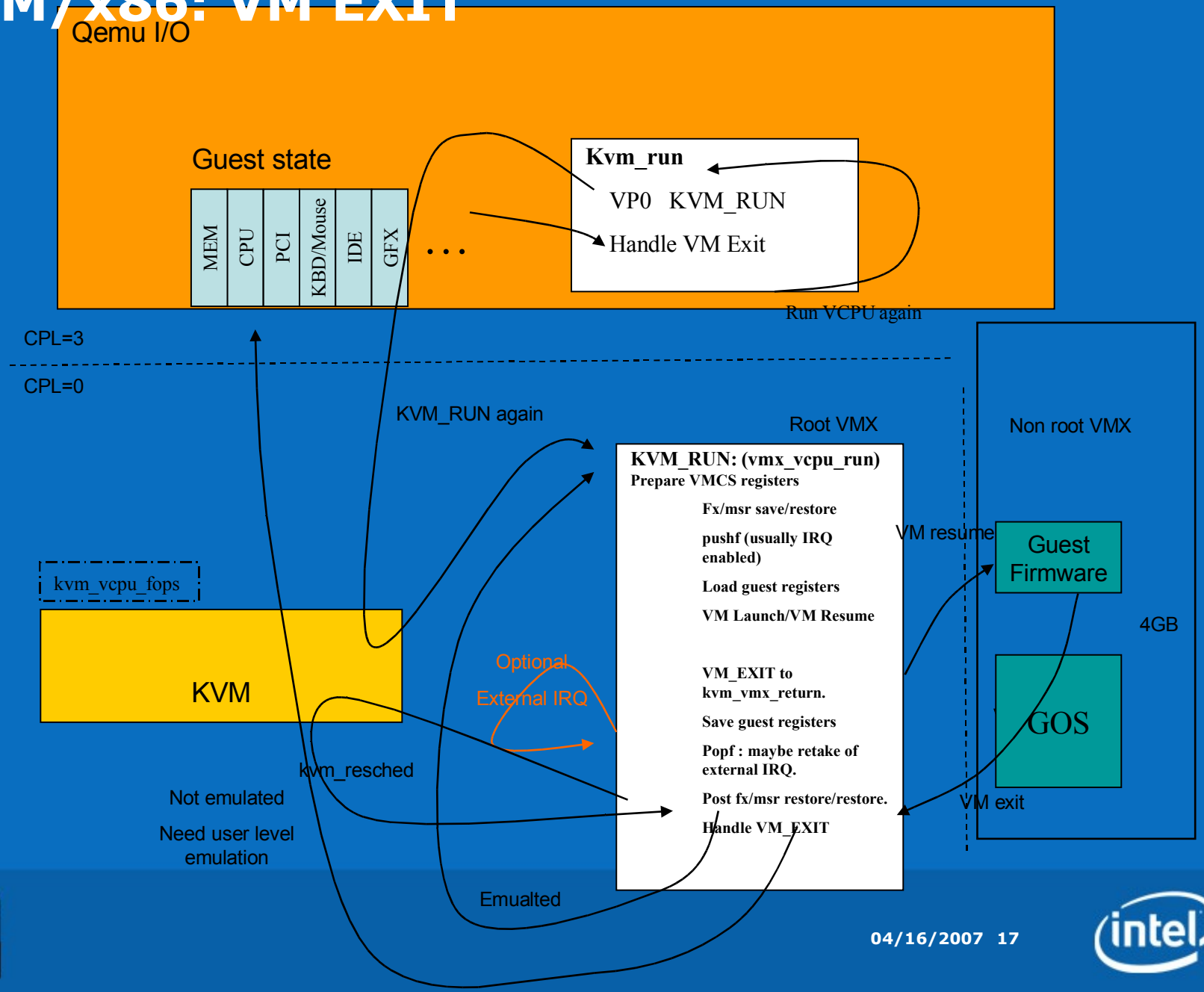
# Q&A

# Backup

# Hardware Virtualization Technologies

- **VT-x: Intel Virtualization Technology for IA-32**
  - Processor emulation by hardware
  - VMx: virutal machine extention
  - Non-root operation and root operation
    - VM entry: transition from root operation to non-root operation
      - VMLAUNCH and VMRESUME instructions
    - VM exit: transition from non-root operation to root operation
      - Certain instructions: CPUID, INVD, mov to CR3, etc.
      - External interrupt
      - No-maskable interrupt (NMI)
      - INITs
      - Exception
      - Startup IPI
      - Task switch
  - VMCS: 4KB memory for VMx states between root and non-root operations.
- VT-i: Intel Virtualization Technology for IA64
  - Processor emulation by hardware
  - VMx: virutal machine extention
  - VM mode (psr.vm=1) and non-VM mode (psr.vm=0)
    - Create VM:
      - PAL_VP_CREATE
      - PAL_VP_RESTORE and PAL_VP_SAVE
    - VM entry: transition from VM mode to non-VM mode
      - PAL_VPS_RESUME_NORMAL/PAL_VPS_RESUME_HANDLE
    - VM exit: transition from non-VM mode to VM mode
      - All privileged instructions
      - Some non-privileged instructions: thash, ttage, cover, etc.
      - External interrupts
      - Interruptions: TLB miss, General Exception, VHPT miss, etc.
  - VPD: 64KB memory for VMx states between VM and non VM modes.

# KVM/x86: VM EXIT




intel®
Leap ahead™